ฉัตรชัย ศิริรัมย์*

เรียนรู้ภาษา โพสต์สคริปต์ ตอนที่ 2

เน้นในส่วนของโอเปอเรเตอร์ต่าง ๆ ที่สามารถนำไปประยุกต์ใช้ในการสร้างภาพกราฟิกที่ต้องการ

จากตอนที่แล้วเราได้กล่าวถึงภาษาโพสต์สคริปต์ในส่วนต่าง ๆ ตั้งแต่ประวัติความเป็นมาของภาษา อุปกรณ์ใดบ้างที่สนับสนุนการเขียนโปรแกรมภาษานี้ พื้นฐานต่าง ๆ ที่สำคัญ อาทิเช่น หลักการการทำงานของสแต็ก ซึ่งก็ได้มีการแทรกการเขียนโปรแกรมด้วยภาษาโพสต์สคริปต์อย่างง่าย ๆ ประกอบเพื่อให้ผู้อ่านได้เข้าใจ และสามารถนำไปใช้เป็นแนวทางในการศึกษาภาษาโพสต์สคริปต์ระดับสูงต่อไป ซึ่งเนื้อหาของบทความตอนที่ 2 นี้ก็จะเน้นในส่วนของโอเปอเรเตอร์ต่าง ๆ ซึ่งสามารถนำไปประยุกต์ใช้ในการสร้างภาพกราฟิกที่ต้องการได้ และก็เช่นเดิมที่จะต้องมีการยกตัวอย่างการเขียนโปรแกรม โดยในฉบับนี้มีตัวอย่างของโปรแกรมภาษาโพสต์สคริปต์ค่อนข้างมาก และหากเป็นไปได้ตัวของท่านผู้อ่านเองก็ควรทำการหัดเขียนโปรแกรมตามตัวอย่างไปด้วย เพราะผู้เขียนมีความเชื่อว่าการที่คนเราจะมีความชำนาญในเรื่องใดก็ตาม โดยเฉพาะในการใช้ภาษาโปรแกรมทางคอมพิวเตอร์ ย่อมเกิดจากการสะสมประสบการณ์ และจากการฝึกฝนจนชำนาญนั่นเอง หากท่านผู้อ่านบางท่านที่ยังไม่เคยได้ศึกษาภาษาโพสต์สคริปต์มาก่อน หรืออยากจะรื้อฟื้นความรู้เดิมเสียก่อน ก็ขอย้อนกลับไปอ่านหนังสือไมโครคอมพิวเตอร์ฉบับที่ 100 ในหัวข้อเรื่อง เรียนรู้ภาษาโพสต์สคริปต์ ตอนที่ 1 จะทำให้ท่านผู้อ่านเข้าใจได้ดียิ่งขึ้น

โดยทั่วไปภาษาที่ใช้ในการติดต่อสื่อสารระหว่างคอมพิวเตอร์กับเครื่องพิมพ์เลเซอร์จะมีอยู่ด้วยกัน 2 ประเภทก็คือ Page Command Language (PCL) และภาษาโพสต์สคริปต์ ซึ่งยังเป็นภาษาที่ไม่แพร่หลายมากนัก นอกจากนี้เราก็ยังสามารถจะนำภาษาโพสต์คริปต์มาใช้ร่วมกับภาษาระดับสูงอื่น ๆ ได้ เพื่อใช้ในการจัดการและควบคุมเครื่องพิมพ์เลเซอร์ได้เช่นกัน ดังนั้นจึงนับได้ว่าเป็นภาษาที่มีประสิทธิภาพมากทีเดียว แต่ขณะนี้ผู้ใช้น้อยคนนักที่สามารถใช้ความสามารถของเครื่องพิมพ์แบบโพสต์สคริปต์ได้อย่างคุ้มค่าเท่าที่ควร นอกจากซอฟต์แวร์บางชนิดที่จะทำหน้าที่ใช้ภาษาโพสต์สคริปต์ในการควบคุมเครื่องพิมพ์แทนที่เราจะต้องมาจัดการเอง

แต่ตอนนี้เรามาเริ่มศึกษาภาษาโพสต์สคริปต์พร้อมตัวอย่างกันก่อนดีกว่านะครับ ซึ่งผมจะไม่ขออธิบายโปรแกรมอย่างละเอียด เนื่องจากว่าในบทความตอนที่แล้วได้อธิบายส่วนของโอเปอเรเตอร์ต่าง ๆ ในโปรแกรมอย่างละเอียดทีละขั้นตอนไปแล้ว จึงไม่ขอกล่าวซ้ำอีก นอกจากส่วนที่ยังไม่เคยได้เขียนอธิบายในตัวอย่างคราวก่อน

*ภาควิชาคณิตศาสตร์และวิทยาการคอมพิวเตอร์ คณะวิทยาศาสตร์ประยุกต์ สถาบันเทคโนโลยีพระจอมเกล้าพระนครเหนือ

อาร์กิวเมนต์	โอเปอเรเตอร์	ผลลัพธ์	หมายเหตุ
num1 num2	add	sum	บวกค่า num1 และ num2
num1 num2	div	quotient	หาร num1 ด้วย num2
int1 int2	idiv	quotient	หารเลขจำนวนเต็ม
int1 int2	mod	remainder	เศษเหลือจากการหาร int1 ด้วย int2
num1 num2	mul	product	num1 คูณกับ num2
num1 num2	sub	difference	ลบ num1 จาก num2
num1	abs	num2	ค่า absolute ของ num 1
num1	neg	num2	ค่า negative ของ num1
num1	round	num2	ปัดค่า num1 ให้เป็นจำนวนเต็ม
num1	truncate	num2	ตัดเศษทศนิยมของ num1 ออก
num	sqrt	real	ค่า square root ของ num
num den	atan	angle	ค่า arctangent ของ num/den
angle	cos	real	ค่า cosine ของ angle
angle	sin	real	ค่า sine ของ angle
base exponent	exp	real	หาค่า exponont ของ base
num	in	real	หาค่า natural logarithm (base e)
num	log	real	หาค่า logarithm (base 10)

ตารางที่ 1
โอเปอเรเตอร์ที่เกี่ยวกับคณิตศาสตร์

การคำนวณทางคณิตศาสตร์

ภาษาโพสต์สคริปต์นั้นได้เอื้ออำนวยโอเปอเรเตอร์ที่จัดการทางด้านคณิตศาสตร์มาให้มากพอสมควร ดังได้แสดงไว้ในตารางที่ 1 ซึ่งก็คงจะเป็นคำสั่งที่คุ้นเคยกันดีสำหรับผู้ที่เคยมีประสบการณ์ในการเขียนโปรแกรมคอมพิวเตอร์ในภาษาอื่น ๆ มาแล้ว ซึ่งจะขอไม่กล่าวอธิบายการนำไปใช้งานอย่างละเอียด เนื่องด้วยเป็นโอเปอเรเตอร์ที่ทำความเข้าใจได้ไม่ยากนัก

การสร้างพาธและการลงสี

การสร้างพาธนั้นก็คือเส้นทางเดินของ current point ที่เป็นเส้นตรง หรืออาจจะเป็นเส้นโค้ง โดยพาธหรือเส้นดังกล่าวเกิดจากการใช้โอเปอเรเตอร์ของ lineto, arcto, curveto operator และโอเปอเรเตอร์อื่น ๆ ที่ใช้ในการลากเส้นโค้งหรือเส้นตรง จากตัวอย่างที่ 1 เป็นการลากเส้นตรง 1 เส้น เมื่อได้พาธแล้ว เราก็สามารถตีเส้นได้โดยใช้ stroke operator แต่หากว่า ณ จุดนี้เราต้องการลงสีให้แก่รูป มันจะยังไม่สามารถที่จะลงสีได้ (ในที่นี้หมายถึงการระบายสีแก่ผลลัพธ์ และการลงสีนั้นเราจะต้องทำการปิดพาธก่อน เพื่อไม่ให้การลงสีนั้นออกนอกขอบเขตของพาธที่กำหนดไว้ จึงเป็นลักษณะของผลลัพธ์หรือภาพที่เป็นรูปทรงต่าง ๆ เช่น รูปทรงทางเรขาคณิตต่าง ๆ ซึ่งการลงสีโดยใช้ fill operator จะแสดงไว้ในตัวอย่างที่ 2

กฎของการลงสี (even-odd rule) ในพาธ

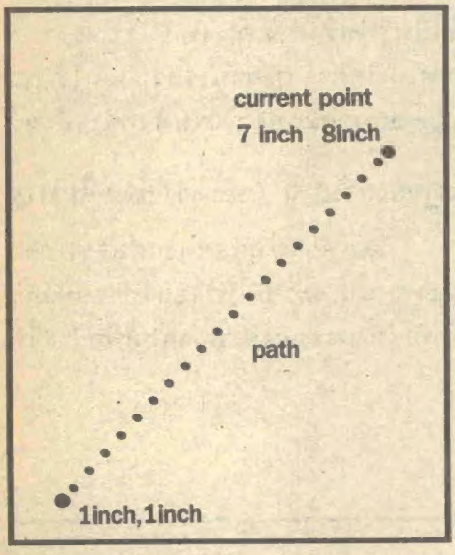
โดยทั่วไปพาธของแต่ละรูปทรงต่าง ๆ นั้น จะมีทิศทางของพาธอยู่แล้ว อาทิเช่น พาธของภาพวงกลมที่สร้างขึ้น

ตัวอย่างที่ 1 การสร้างพาธและตีเส้น

```
%!PS
/inch {72 mul} def                  % define inch procedure
1 inch 1 inch moveto                % set current point
7 inch 8 inch lineto                % line segment
stroke                              % paint the path
showpage                            % display page
```

ตัวอย่างที่ 2 พาธและการลงสีแบบซ้อนทับกัน

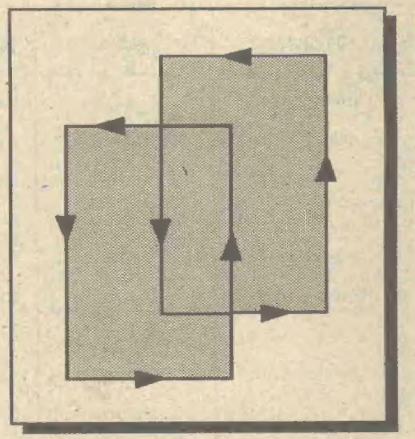
```
%!PS
/inch {72 mul} def                  % define inch procedure
/width 4 inch def                   % define width of rectangle
/height 7 inch def                  % define height of rectangle
1 inch 1 inch moveto                % set current point
width 0 rlineto                     % bottom of rectangle
0 height rlineto                    % right side
width neg 0 rlineto                 % top
closepath                           % close the path
7.5 inch 10 inch moveto             % set current point
width neg 0 rlineto                 % top of rectangle
0 height neg rlineto                % left side
width 0 rlineto                     % bottom
closepath                           % close the path
0.50 setgray                        % set fill shade
fill                                % fill the shapes
showpage                            % display page
```



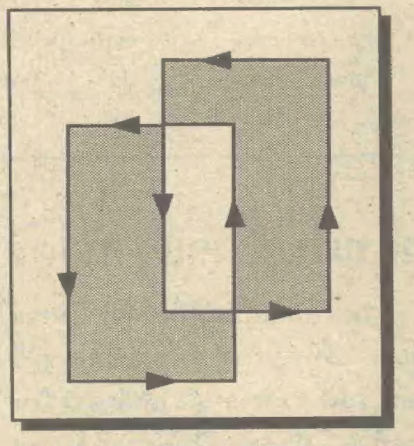
รูปที่ 1 ผลลัพธ์การสร้างพาธ

มา พาธของมันจะมีทิศทวนเข็มนาฬิกา ถ้าเราสร้างวงกลมด้วย arc operator แต่ถ้าเราสร้างวงกลมด้วย arcn operator จะทำให้พาธของมันจะมีทิศตามเข็มนาฬิกา ซึ่งถ้าเกิดว่าเราสร้างภาพกราฟิกที่เป็นรูปทรงใด ๆ 2 รูป และมีการลงสีที่มีการซ้อนทับกันของภาพดังกล่าวทั้งคู่ โดยมันจะทำให้การลงสีเป็นไปตามกฎเกณฑ์ดังนี้คือ ถ้าทิศทางของพาธด้านในและด้านนอกสวนทางกัน จะเกิดช่องว่างที่ไม่ได้รับการเติมสีลงไปในพาธนั้น แต่ถ้าพาธทั้งสองนั้นมีทิศทางในทางเดียวกัน พาธจะถูกลงสีซ้อนทับกันตามลำดับก่อนหลังของพาธที่ถูกสร้างขึ้น และในกรณีที่พาธนั้นซ้อนทับพาธอื่นอย่างหมดสิ้น จะทำให้พาธที่ใหญ่กว่าถูกระบายสีเท่านั้น ซึ่งการลงสีในกรณีนี้เราต้องใช้ eofill operator

ในการลงสีในพาธดังตัวอย่างที่ 2 เป็นการสร้างพาธที่ซ้อนทับกันแบบมีทิศทางของพาธตามกัน พร้อมกับการลงสี ซึ่งผลลัพธ์จะได้แสดงดังรูปที่ 2



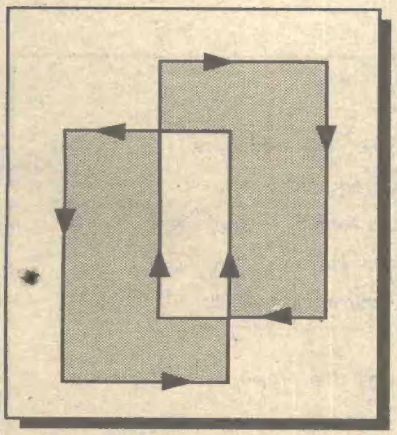
รูปที่ 2 ผลลัพธ์ของพาธและการลงสีแบบซ้อนทับกัน



รูปที่ 3 ผลลัพธ์จากการใช้ eofill operator

จากตัวอย่างที่ 2 จะเห็นว่าการลงสีนั้นเกิดการซ้อนทับกันเมื่อพาธมีทิศทางตามกัน ถ้าเราลองมาเปลี่ยนตัวอย่างที่ 2 จากเดิมที่ได้ใช้ fill operator เราจะหันมาใช้ eofill operator แทน ซึ่งจะได้ผลลัพธ์ดังรูปที่ 3

ในกรณีที่พาธที่เราสร้างขึ้นมีทิศทางที่สวนทางกัน จะให้ผลลัพธ์เช่นเดียวกับที่เราใช้ eofill operator นั่นเอง ซึ่งจะแสดงดังรูปที่ 4 โดยเราจะใช้ fill operator ดังเดิม แต่จะเปลี่ยนทิศทางของพาธอันแรกให้มีทิศทางกลับจากเดิม



รูปที่ 4 ผลลัพธ์การสร้างพาธที่มีทิศทางสวนกัน

ทรานฟอร์เมชัน

การทรานฟอร์เมชันคือเป็นการเปลี่ยนแปลงรูปแบบของระบบโคออร์ดิเนตซึ่งเคยได้กล่าวและยกตัวอย่างไปบ้างในบทความตอนที่ 1 การทรานฟอร์เมชันนี้สามารถนำไปใช้ในการเปลี่ยนแปลงรูปแบบภาพที่เราสร้างขึ้นมาได้หลายหลาก โดยการทรานฟอร์เมชันจะมีการกระทำกับโคออร์ดิเนตอยู่ด้วยกัน 3 ประเภทดังนี้

● **การย้ายจุด origin** (translating the origin) โดยใช้ translate operator ซึ่งต้องการอาร์กิวเมนต์ 2 ตัวเป็นตัวบ่งบอกจุด origin ใหม่ และจะมีรูปแบบการใช้งานดังนี้

```
Tx Ty translate
```

● **การเปลี่ยนสเกล** (scaling the coordinate system) หมายถึงเป็นการย่อ-ขยายขนาดของสเกลได้ตามแนวแกน x และแกน y โดยใช้ scale operator ซึ่งจะต้องการอาร์กิวเมนต์ 2 ตัวเป็นตัวบ่งบอกสเกลใหม่ ตามหน่วยมาตรฐานของแกน x และแกน y จะมีรูปแบบการใช้งานดังนี้

```
Sx Sy scale
```

● **การหมุนแกน** (rotating the coordinate system) โดยใช้ rotate operator ต้องการอาร์กิวเมนต์ที่เป็นมุมในการหมุนแกน จะมีรูปแบบการใช้งานดังนี้

```
angle rotate
```

Current Transformation Matrix (CTM)

เป็นแมตริกซ์ที่ประกอบไปด้วยสมาชิก 6 ตัว ซึ่งเป็นแมตริกซ์ที่ทำหน้าที่เปลี่ยนจุด ณ ตำแหน่งบนระบบโคออร์ดิเนตที่ผู้ใช้ใช้ลงจุดบนเอาต์พุตเพจ หรือที่เราเรียกว่า user space (ใน 1 จุดจะมีขนาดเป็น 1/72 หน่วยมาตรฐานของโพสต์สคริปต์) ให้เป็นจุดที่แท้จริง (pixel) ในระบบโคออร์ดิเนตของอุปกรณ์แต่ละชนิด ซึ่งเราจะเรียกว่า device space เราจะสามารถเขียน CTM ในรูปของสมการได้ดังนี้

$$X_{device} = aX_{user} + bY_{user} + T_x$$
$$Y_{device} = cX_{user} + dY_{user} + T_y$$

ถ้าจะเขียน CTM ให้อยู่ในรูปของแมตริกซ์จะได้ดังนี้

$$[a\ b\ c\ d\ T_x\ T_y]$$

ซึ่งโดยทั่วไป CTM จะมีค่าดังนี้

```
[1 0 0 1 0 0]
```

จากหลักการดังที่ได้กล่าวมาแล้ว เราจะสามารถนำ CTM ไปประยุกต์ใช้งานต่าง ๆ ได้ดังนี้

การเคลื่อนย้ายจุด origin ซึ่งการเคลื่อนย้ายจุด origin โดยปกติแล้วเราจะใช้ translate operator แต่ในที่นี้เราจะมาประยุกต์การ translate ด้วยการเปลี่ยน CTM โดยใช้ concat operator และเราจะมาดูถึงหลักการเคลื่อนย้ายจุด origin กันก่อน เพื่อที่จะสามารถนำไปประยุกต์ใช้ได้ ส่วนการเคลื่อนย้ายจุด origin จะเกิดจากสมการดังนี้

$$X_{new} = X_{original} + T_x$$
$$Y_{new} = Y_{original} + T_y$$

เพราะฉะนั้นการ translate โดยค่าของ CTM จะถูกใช้งานด้วย concat operator และมีสมาชิกของแมตริกซ์ดังต่อไปนี้

$$[1\ 0\ 0\ 1\ T_{original\,x} + T_x\ T_{original\,y} + T_y]\ \text{concat}$$

การเปลี่ยนสเกล แต่เดิมการเปลี่ยนสเกลเราจะใช้ scale operator แต่ในที่นี้เราจะมาเปลี่ยนโดยใช้ CTM ซึ่งจะใช้สมการดังต่อไปนี้

$$X_{new} = X_{original} \times S_x$$
$$Y_{new} = Y_{original} \times S_x$$

และสามารถจะจัดให้อยู่ในรูปของโอเปอแรนด์ของ concat operator ได้ดังนี้

$$[S_x\ 0\ 0\ S_y\ 0\ 0]\ \text{concat}$$

การ rotate แกน การ rotate นั้นจะต้องอาศัย rotate operator และถ้าใช้ CTM ก็จะอยู่ในรูปของสมการดังนี้

$$X_{new} = X_{original} \times \text{cosine} - Y_{original} \times \text{sine}$$
$$Y_{new} = X_{original} \times \text{sine} + Y_{original} \times \text{cosine}$$

และสามารถจัดให้อยู่ในรูปเมตริกซ์ของ concat operator ดังนี้

```
[cosine -sine sine cosine 0 0]concat
```

การแสดงข้อความ

งานทางด้านกราฟิกที่นิยมนำเอาภาษาโพสต์สคริปต์มาใช้ก็คือ งานทางด้านแสดงอักขระข้อความต่าง ๆ อาจจะเป็นโลโกเพื่อใช้แทนสัญลักษณ์ หรืออาจจะเป็นงานด้านอื่น ๆ ที่ต้องใช้การแสดงอักขระ ซึ่งภาษาโพสต์สคริปต์สามารถพลิกแพลงและตกแต่งผลลัพธ์ได้อย่างดีเยี่ยม

การแสดงตัวอักษรหรืออักขระนั้น เราต้องนำข้อความที่จะแสดงให้อยู่ในเครื่องหมายวงเล็บเปิดและปิด และปิดท้ายด้วย show operator ซึ่งเราจะได้กล่าวถึงลำดับขั้นตอนของการแสดงผลตัวอักษรดังนี้

- ติดตั้ง current font ซึ่งจะประกอบด้วยชนิดของฟอนต์และขนาดตามที่ต้องการลงสู่กราฟิกสเตท
- ข้อความจะถูกแทนลงไปยัง current point
- ข้อความจะปรากฏบนระบบโคออร์ดิเนตของ user space
- ข้อความจะถูกแสดงตาม current color
- ข้อความจะถูกเปลี่ยนให้อยู่ในรูปกราฟิกออปเจ็กต์โดยใช้ CTM

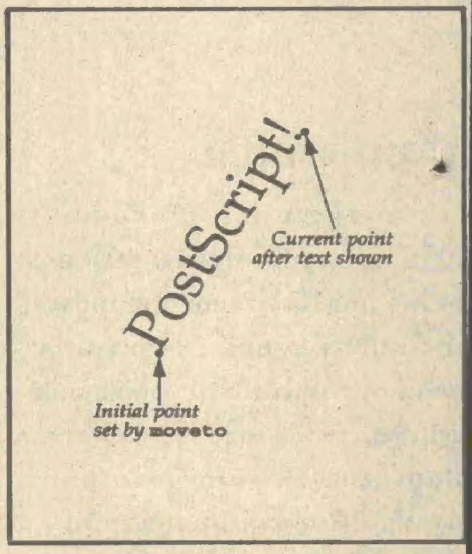
โดยทั่วไปเราจะเลือกใช้โอเปอเรเตอร์ที่เป็นชุดของการแสดงข้อความที่เป็นพื้นฐานคือ findfont, scale และ setfont operator ซึ่งจะแบ่งออกเป็น 3 ขั้นตอนดังนี้

- ค้นหาฟอนต์ที่ต้องการ (finding)
- ติดตั้งขนาดของฟอนต์ที่ต้องการ (scaling)
- เซตรูปแบบฟอนต์ลงสู่กราฟิกสเตท (setting)

ดังจะได้แสดงในตัวอย่างที่ 3 เป็นการแสดงถึงพื้นฐานการแสดงข้อความอย่างง่าย ๆ และรูปที่ 5 เป็นการแสดงผลลัพธ์ของตัวอย่าง 3 และแสดงถึงจุดที่เปลี่ยนแปลงหลังจากแสดงข้อความที่ต้องการแล้ว

ตัวอย่างที่ 3 การแสดงข้อความ

```
%IPS
/inch {72 mul} def                  % define inch procedure
/Palatino-Roman findfont            % find required font
2.25 inch scalefont                 % scale font to required size
setfont                             % set as current font
1.5 inch 0.5 inch moveto            % set current point
52 rotate
(PostScript) show                   % image title on page
showpage                            % display page
```



รูปที่ 5 ผลลัพธ์ของการแสดงข้อความ

การ transforming font โดยใช้แมตริกซ์

การติดตั้งฟอนต์ที่เราได้กล่าวมาแล้วนั้น ล้วนแล้วแต่เป็นการติดตั้งฟอนต์ในแบบพื้นฐานที่ง่าย ๆ เท่านั้น แต่ในที่นี้เราจะมาประยุกต์การติดตั้งฟอนต์โดยใช้ฟอนต์เมตริกซ์ ซึ่งจะเป็นเมตริกซ์ที่เป็นข้อมูลสำหรับใช้ในการแปลงอักขระในระบบ character coordinate system ให้เป็นอักขระในระบบ user coordinate system โดยอาศัยหลักการเช่นเดียวกับ CTM และเราจะใช้แทนการติดตั้งขนาดของฟอนต์นั่นเอง โดยอาศัย makefont operator โดยโอเปอเรเตอร์นี้ต้องการอาร์กิวเมนต์ที่เป็นเมตริกซ์เพื่อใช้ในการกำหนดทรานฟอร์ม ดังตัวอย่างที่ 4 แสดงถึงการทรานฟอร์มฟอนต์ และรูปที่ 6 เป็นผลลัพธ์ของตัวอย่างที่ 4

การแสดงข้อความภายใต้โพรซีเดอร์คอนโทรล

หมายถึงการแสดงข้อความในลักษณะลูปตามจำนวนครั้งของจำนวนอักษรทั้งหมดลบด้วยหนึ่ง (หมายถึงถ้าจำนวนอักษรในข้อความนั้นมี 5 ตัวอักษร จะทำให้การทำลูปดังกล่าวเกิด

ตัวอย่างที่ 4 การทรานฟอร์มฟอนต์

```
%!PS
/inch {72 mul} def                          % define inch procedure
/ZapfChancery-MediumItalic findfont         % find desired font
[10 inch 0 0 10 inch 0 0] makefont          % scale it large
setfont                                     % set font in graphics state
-0.5 3 inch moveto                          % set current point
(Q) show                                    % image the character
showpage                                    % display page
```

รูปที่ 6 ผลลัพธ์ของการทรานฟอร์มฟอนต์

ขึ้น 4 ครั้ง) ซึ่งจะปฏิบัติตามคำสั่งในโพรซีเดอร์ที่ควบคุมการแสดงผลของแต่ละอักษรนั้น เราจะใช้ kshow operator ซึ่งโอเปอเรเตอร์นี้ต้องการอาร์กิวเมนต์ 2 ตัวก็คือ โพรซีเดอร์กับสตริง ซึ่งในแต่ละรอบของการทำซ้ำ จะมีการใส่อักษรตัวแรกและตัวถัดไปลงในสแต็ก ดังจะได้แสดงในตัวอย่างที่ 5 การแสดงผลของอักษรในข้อความที่มีเฉดสีที่แตกต่างกัน และรูปที่ 7 จะแสดงผลลัพธ์ของตัวอย่างที่ 5

การสร้างพาธจากตัวอักษร

ตัวอักษรต่าง ๆ ที่ได้แสดงนั้น

ตัวอย่างที่ 5 การแสดงอักษรด้วยโพรซีเดอร์คอนโทรล

```
%!PS
/inch {72 mul} def                          % define inch procedure
/Palatino-BoldItalic findfont               % choose font
2.5 inch scalefont setfont
/gray 0.1 def                               % gray for second character
1.125 inch 0.875 inch moveto                % set current point
52 rotate                                   % turn coordinate system
{pop pop                                    % pop character off stack
  gray setgray                              % change gray shade
  /gray gray 0.1 add def                    % increment for next time
} (fading out) kshow                        % show string
showpage                                    % display page
```

รูปที่ 7 ผลลัพธ์ของการแสดงผลด้วยโพรซีเดอร์คอนโทรล

เราสามารถจะสร้างพาธล้อมรอบตัวอักษรได้ และนำไปใช้เช่นเดียวกับในเรื่องของการสร้างพาธและการลงสี ที่เราได้กล่าวถึงไปแล้วในตอนต้น โดยอาศัย charpath operator ซึ่งต้องการอาร์กิวเมนต์ 2 ตัวที่เป็นสตริงและค่า boolean ส่วนค่าของ boolean นั้น ถ้าเป็นจริง (true) จะใช้สำหรับการลงสี แต่ถ้าเป็นเท็จ (false) จะใช้สำหรับการตีเส้น ดังตัวอย่างที่ 6 จะเป็นการตีเส้นล้อมรอบตัวอักษร ทำให้ได้ตัวอักษรที่เป็นลักษณะแบบเอาต์ไลน์ฟอนต์ และรูปที่ 8 เป็นผลลัพธ์ของตัวอย่างที่ 6

การวาดเส้นตรงและเส้นโค้ง

การวาดเส้นตรงในภาษาโพสต์สคริปต์นั้นกระทำได้ง่ายมาก โดยใช้เพียง lineto และ rlineto โอเปอเรเตอร์เท่านั้น โดยเป็นการวาดจากจุด current point ไปยังจุด x, y ที่เป็นอาร์กิวเมนต์ของโอเปอเรเตอร์ทั้งสอง แต่ rlineto

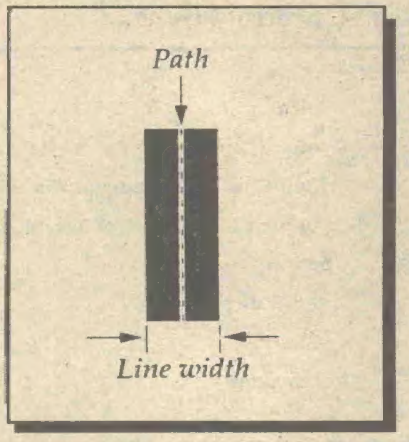
ตัวอย่างที่ 6 การตีเส้นล้อมรอบอักษร

```
%IPS
/inch {72 mul} def                          % define inch procedure
/ZapfChancery-MediumItalic findfond % find required font
10 inch scalefont                           % scale it to fit page
setfont                                     % set font in graphics state
0.5 inch 3 inch moveto                      % set current point
(Z) false charpath                          % obtain character path
0.0 setgray                                 % set black color
8 setlinewidth                              % set fat line width
stoke                                       % paint the path
showpage                                    % display page
```

รูปที่ 8 ผลลัพธ์การตีเส้นล้อมรอบอักษร

จะแตกต่างตรงที่ว่าอาร์กิวเมนต์ที่ให้มาจะใช้บวกเพื่อเพิ่มค่าให้กับ current point (ได้กล่าวอธิบายในตัวอย่างที่มีการใช้ lineto และ rlineto operator ไปแล้วในฉบับก่อน) ข้อควรระวังในการลากเส้นตรงหรือเส้นโค้งก็คือ เราจำเป็นจะต้องมี current point ก่อน มิฉะนั้นจะเกิดข้อผิดพลาดออกมาว่า nocurrentpoint การได้เส้นตรงหรือเส้นโค้ง 1 เส้น เราจะถือว่าเกิดพาธขึ้นมา 1 พาธ ซึ่งสามารถจะนำไป stroke หรือตีเส้นทึบเพื่อเป็นการเน้นเส้นนั้น ๆ ได้

ส่วนการเซตขนาดของเส้นตรงนั้น เราจะใช้ setlinewidth operator ซึ่งต้องการอาร์กิวเมนต์เพียงตัวเดียวที่เป็นตัวบ่งบอกถึงขนาดของเส้นตรง โดย 1 หน่วยของมันมีค่าเท่ากับ 1/72 นิ้ว สมมติว่าเราต้องการให้เส้นที่จะทำการ stroke นั้นมีขนาดเท่ากับ 0.5 นิ้ว เราก็จะกำหนดค่าของอาร์กิวเมนต์เป็น 36 และเราจะไม่สามารถที่จะเซตขนาดของเส้นตรงให้มีขนาดต่ำกว่าหรือเท่ากับ 0 ได้ รูปที่ 9 แสดงถึงความสัมพันธ์ระหว่างพาธที่เกิดจากการวาดเส้นตรงและความกว้างของเส้นตรงที่เราได้เซตไว้

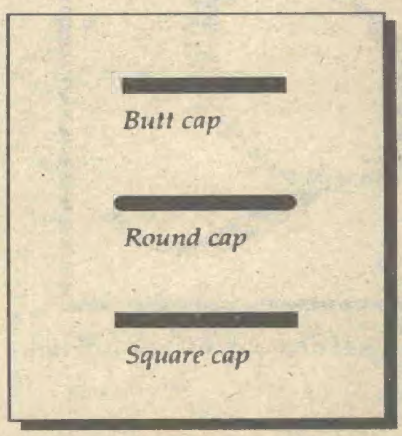


รูปที่ 9 ความสัมพันธ์ระหว่างความกว้างของเส้นตรงและพาธ

line caps

หมายถึงการกำหนดรูปแบบของจุดสิ้นสุดของเส้นตรงที่เกิดจากการ stroke โดยแบ่งออกเป็น 3 ประเภท ดังรูปที่ 10 ซึ่งเราสามารถจะเซตได้โดยใช้ setlinecap operator และต้องการอาร์กิวเมนต์ 1 ตัว จะมีรูปแบบดังนี้

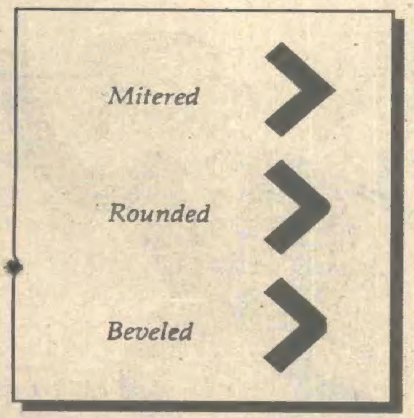
- 0 setlinecap เซตให้เป็น butt cap ซึ่งเป็นค่าดีฟอลต์ ถ้าหากเราไม่ได้ setlinecap ใหม่
- 1 setlinecap เซตให้เป็น round cap
- 2 setlinecap เซตให้เป็น square cap



รูปที่ 10 ชนิดของ line caps

line joins

หมายถึงการกำหนดรูปแบบของเส้นตรง 2 เส้น ที่มาเชื่อมต่อกัน ซึ่งมีอยู่ด้วยกัน 3 รูปแบบดังได้แสดงในรูป



รูปที่ 11 ชนิดของการเชื่อมต่อเส้นตรง

ที่ 11 โดยเราจะใช้ setlinejoin operator จะมีรูปแบบดังนี้

- 0 setlinejoin เป็นการเชื่อมต่อแบบ Mitered ซึ่งเป็นค่าดีฟอลต์
- 1 setlinejoin เป็นการเชื่อมต่อแบบ round
- 2 setlinejoin เป็นการเชื่อมต่อแบบ beveled

Dashed line

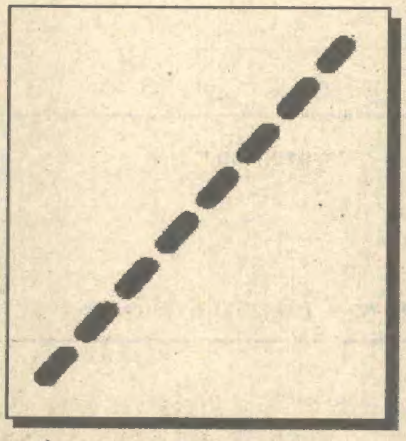
เป็นการทำให้เส้นตรงหรือเส้นโค้งนั้นเป็นเส้นประตามรูปแบบที่เราสามารถจะกำหนดได้ด้วย setdash operator โดยจะมีรูปแบบดังนี้

$[on_1\ off_1\ on_2\ off_2 \ldots on_n\ off_n\ ]$ offset setdash

setdash จะมีอยู่ 2 พารามิเตอร์ คือ อะเรย์ของ on_n และ off_n และออฟเซตแต่ละค่าของ on_n นั้นจะเป็นตัวบ่งบอกจำนวนจุดที่จะแสดงในแนวตามยาวของเส้นตรง ในทางกลับกัน off_n จะเป็นตัวบ่งบอกถึงระยะทางของช่องว่าง (gap) ที่ไม่ถูกแสดงในส่วนของเส้นตรงนั้น ๆ ส่วนออฟเซตนั้นเป็นตัวกำหนดระยะภายในรูปแบบของ dash

ตัวอย่างที่ 7 การวาดเส้นประ พร้อมกับ setlinecap

```
%IPS
/inch {72 mul} def                    % define inch procedure
1 inch 1 inch moveto                  % set current point
7.5 inch 10 inch lineto               % draw line across page
[0.75 inch 0.75 inch] 0 setdash       % dash pattern
0.5 inch setlinewidth                 % fat line width
1 setlinecap                          % round line caps
stroke                                % paint the path
showpage                              % display page
```



รูปที่ 12 ผลลัพธ์การวาดเส้นประ พร้อมกับ setlinecap

ที่เส้นประเส้นแรกจะเริ่มต้น เพื่อให้เข้าใจเกี่ยวกับการใช้ setdash เรามาลองดูตัวอย่างที่ 7 ซึ่งเป็นตัวอย่างของการแสดงผลเส้นประ ซึ่งเส้นประนี้มีรูปแบบการสิ้นสุดของเส้นตรงนั้นเป็นแบบ round โดยเกิดจากการที่เราใช้ setlinecap นั่นเอง และผลลัพธ์จะได้แสดงดังรูปที่ 12

วงกลม

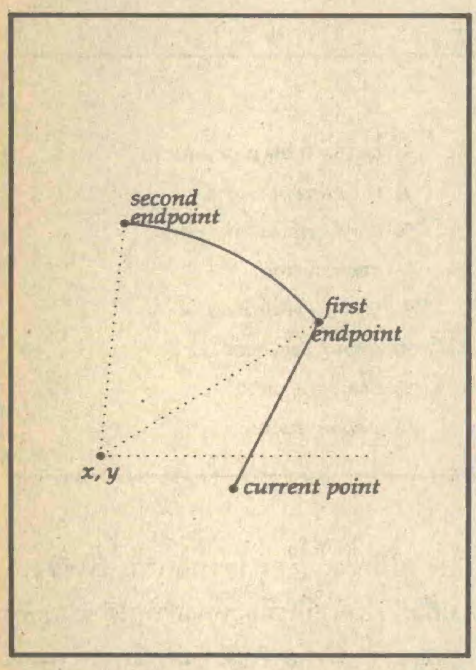
เราได้กล่าวถึงการวาดรูปวงกลมไปบ้างแล้วในฉบับที่แล้ว (ตอนที่ 1) แต่ในที่นี้เราจะมากล่าวถึงลักษณะที่ละเอียดลงไปอีก การสร้างวงกลมนั้นจริง ๆ แล้วเราไม่จำเป็นที่จะต้องกำหนด current point ขึ้นมาก่อน แต่ถ้าเกิดมีการกำหนด current point ขึ้นมาก่อนแล้ว เมื่อมีการสั่งวาดวงกลมจะมีการลากเส้นจาก current point ไปยังจุดที่มีการเริ่มวาดของวงกลม เราลองดูรูปที่ 13 แสดงถึงการกำหนด current point มาก่อนที่จะมีการวาดรูปวงกลม

โดยทั่วไปวงกลมที่เราสร้างจาก arc operator นั้นจะเป็นการวาดวงกลมในทิศทางทวนเข็มนาฬิกา แต่ในภาษาโพสต์สคริปต์นั้นได้มี arcn operator ซึ่งมีทิศทางการวาดเป็นแบบตามเข็มนาฬิกา ซึ่งรูปแบบของอาร์กิวเมนต์มีลักษณะที่คล้ายคลึงกับ arc operator รูปที่ 14 แสดงถึงการกระทำของ arcn operator

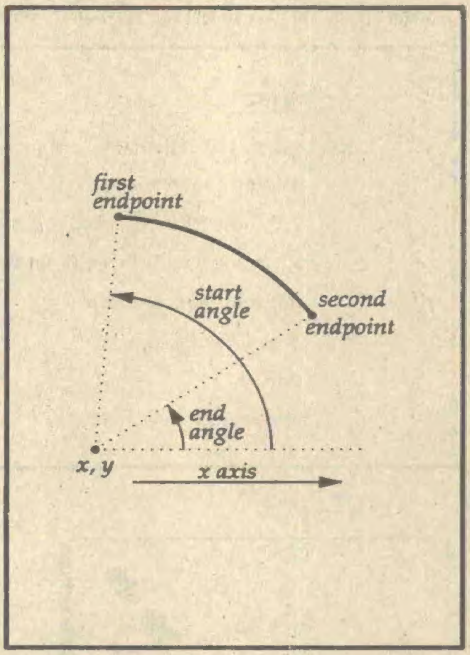
เส้นโค้ง

การวาดเส้นโค้งในภาษาโพสต์-สคริปต์จะเกิดจากสมการของ bezier ซึ่งการวาดเส้นโค้งเราจะใช้ curveto operator โดยมีรูปแบบดังนี้

$X_1Y_1X_2Y_2X_3Y_3$ curveto



รูปที่ 13 ผลลัพธ์ของการวาดวงกลมที่กำหนด current point มาก่อน



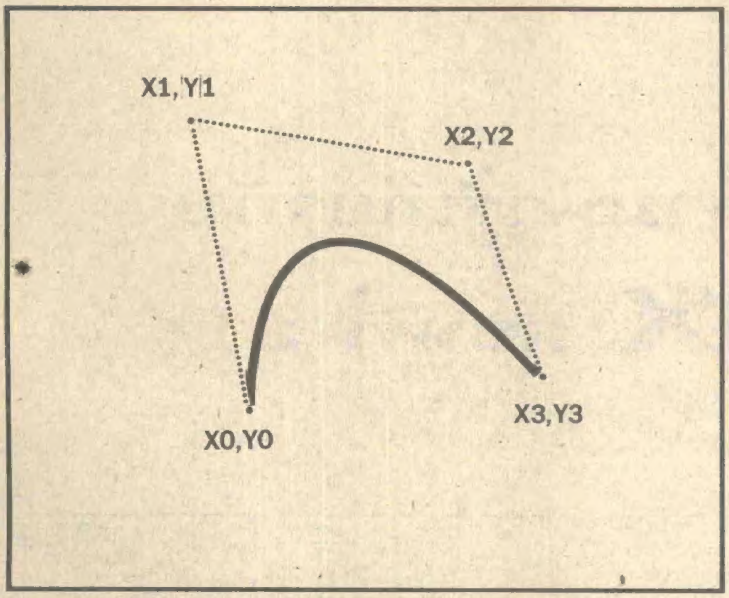
รูปที่ 14 การกระทำของ aren

จุด (x_1, y_1) และ (x_2, y_2) เป็น bezier control point และจะมีจุดสิ้นสุดที่ (x_3, y_3) แต่มีข้อควรระวังไว้ว่าเราจะต้องกำหนด current point ให้เสียก่อนซึ่งในที่นี้เราจะกำหนดให้เป็น (x_0, y_0) การ control point นั้นเราจะแสดงได้ดังรูปที่ 15 และตัวอย่างที่ 8 แสดงการใช้ curveto วาดรูปหัวใจสีชมพู และรูปที่ 16 จะแสดงผลลัพธ์รูปหัวใจสีชมพู

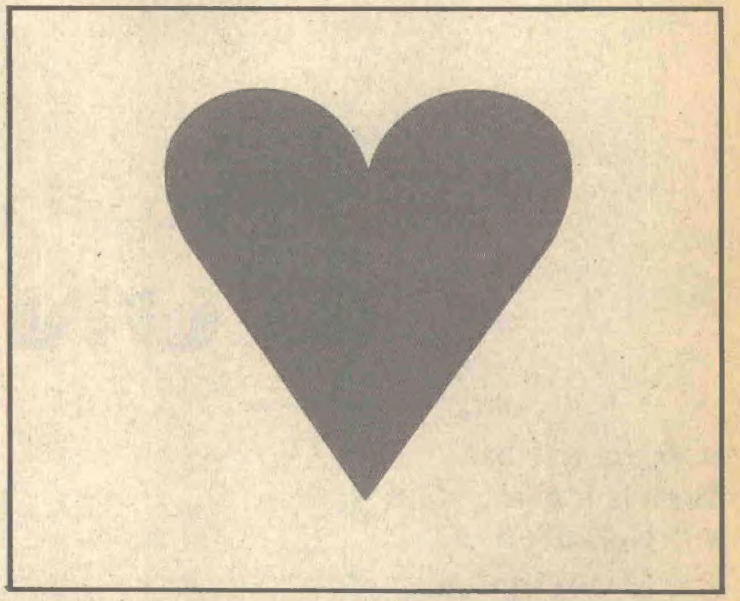
relative bezier curves เป็นการวาดเส้นโค้งโดยใช้ rcurveto operator ซึ่งสามารถวาดเส้นโค้งได้เช่นเดียวกับ curveto แต่ว่า current point ที่เป็นตำแหน่ง x, y ในแต่ละจุดบนเส้นโค้งจะเป็นตำแหน่ง x, y ที่มีลักษณะเป็นค่าสะสม เช่นเดียวกับ rlineto operator ในเรื่องของเส้นตรง อาทิเช่น จุด control point จุดแรกจะกลายเป็น (x_0+dx_1, y_0+dy_1) ซึ่ง rcurveto

ตัวอย่างที่ 8 การวาดรูปหัวใจสีชมพู

```
%IPS
/inch {72 mul} def                      % define inch procedure
/DrawHeart{                             % stack = size
  /size exch def                        % define size temporarily
  /offset size 4 div def                % define offset temporarily
  0 0 moveto                            % first end point
  size size                             % first Bezier control point
  offset size offset add                % second Bezier control point
  0 size                                % second end point
  curveto                               % draw Bezier curve
  offset neg size offset add            % first Bezier control point
  size neg size                         % second Bezier control point
  0 0                                   % second end point
  curveto                               % draw Bezier curve
}def
4.25 inch 1 inch translate              % set origin
8 inch DrawHeart                        % draw heart shape
1.0 0.0 0.5 setrgbcolor fill            % fill with nice pink
showpage                                % display page
```



รูปที่ 15 แสดงการใช้ control point ใน bezier curve



รูปที่ 16 ผลลัพธ์ของการวาดหัวใจสีชมพู

operator จะมีรูปแบบดังนี้

$dx_1\ dy_1\ dx_2\ dy_2\ dx_3\ dy_3$ rcurveto

จากตัวอย่างที่ 9 เราจะเปลี่ยนการใช้ DrawHeart มาใช้โพรซีเดอร์ที่เรากำหนดขึ้นใหม่ โดยใช้ rcurveto operator โดยจะกำหนดไว้ใน procedure RelativeHeart ดังตัวอย่างที่ 9 การวาดรูปหัวใจโดยใช้ relative curve ซึ่งจะให้ผลลัพธ์เช่นเดียวกับตัวอย่างที่ 8 คือรูปที่ 16 นั่นเอง

ตัวอย่างที่ 9 การวาดรูปหัวใจโดยใช้ rcurveto

```
%!PS
/RelativeHeart{                          % stack = size
  /size exch def                         % define size temporarily
  /offset size 4 div def                 % define offset temporarily
  0 0 moveto                             % first end point
  size size offset size offset add 0 size rcurveto
  offset neg offset size neg 0 0 size neg rcurveto
  } def
```

บทสรุป

มาถึง ณ จุดนี้แล้วท่านผู้อ่านคงจะมองเห็นแนวทางในการเขียนโปรแกรมภาษาโพสต์สคริปต์กันบ้างแล้ว ซึ่งโอเปอเรเตอร์ต่าง ๆ ของภาษานี้มีด้วยกันมากมาย ขึ้นอยู่กับว่าเราจะนำโอเปอเรเตอร์ต่าง ๆ ดังกล่าวมาประยุกต์ใช้งานได้อย่างไร ซึ่งก็ขึ้นอยู่กับเทคนิคและความชำนาญของแต่ละบุคคล หากท่านผู้อ่านนึกสนุกขึ้นมา หรือมีความคิดสร้างสรรค์ในการวาดรูป ก็ลองจินตนาการภาพนั้นด้วยภาษาโพสต์สคริปต์ เป็นการท้าทายความสามารถ และฝึกฝนการเขียนโปรแกรมนอกเหนือไปจากตัวอย่างโปรแกรมต่าง ๆ ที่ผมได้สอดแทรกในเนื้อหาของบทความ ซึ่งก็เป็นตัวอย่างโปรแกรมที่เข้าใจได้ไม่ยากนัก และเนื่องด้วยภาษาโพสต์สคริปต์นั้นมีโอเปอเรเตอร์จำนวนมากมาย จึงไม่สามารถที่จะอธิบายได้หมดภายใน 2 ตอน เพราะฉะนั้นผู้เขียนจึงขอยกยอดที่เหลือของภาษาโพสต์สคริปต์ไปไว้ฉบับหน้า ก็อย่าลืมติดตามกันต่อในฉบับหน้านะครับ